Python 101

Deep Learning

DataLab, CS, NTHU

python™

- Readable
- Flexible
- Fast and Powerful
- For this course: Python 3

Outline

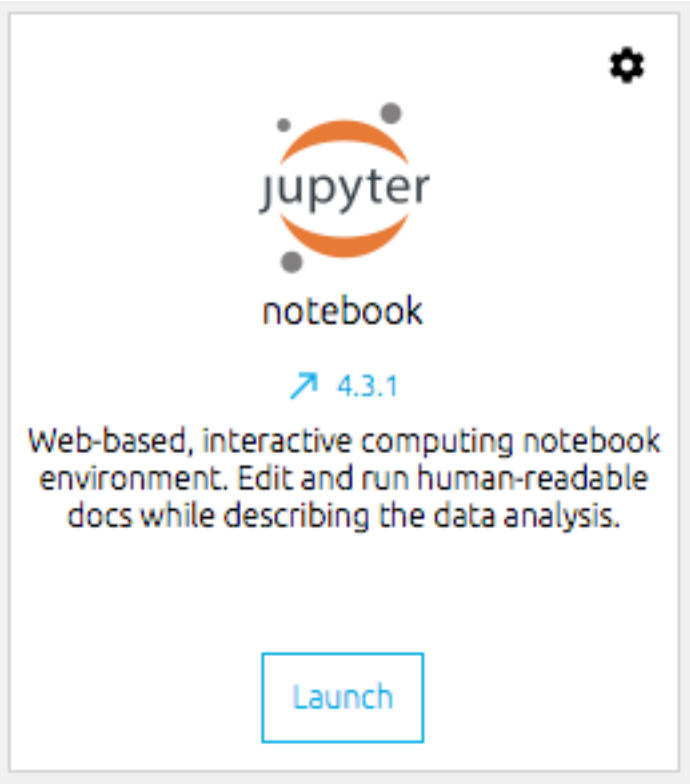
- Environment setup
- Basic Python
- Numpy
- SciPy
- Matplotlib

Environment setup

- Easy way: Install anaconda

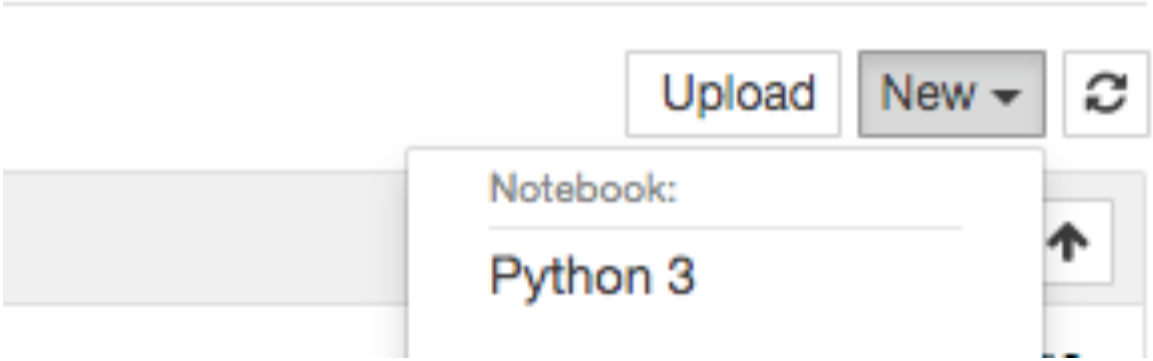
Environment setup

- Launch jupyter notebook inside Anaconda



Environment setup

- Create new notebook



Environment setup

- Inside notebook

```
In [4]: print ("Hello world!")

        Hello world!
```

Environment setup

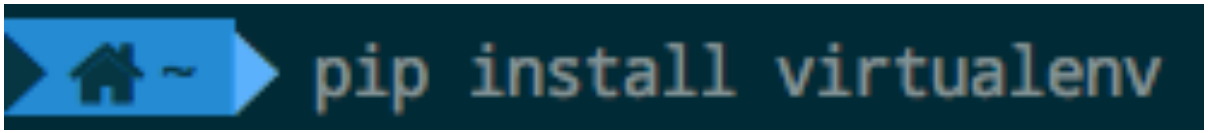
Setting up virtualenv

- Install python
- Install pip

Environment setup

Setting up virtualenv

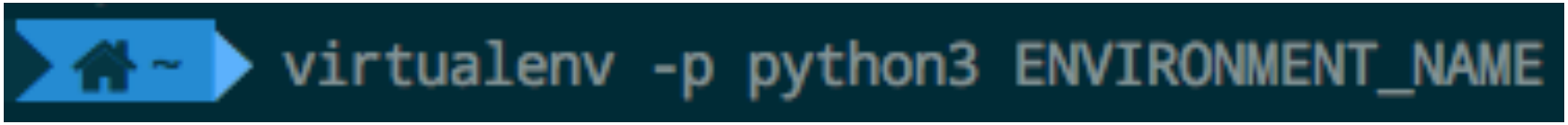
- On the terminal, install virtualenv:



Environment setup

Setting up virtualenv

- Create a virtual environment:

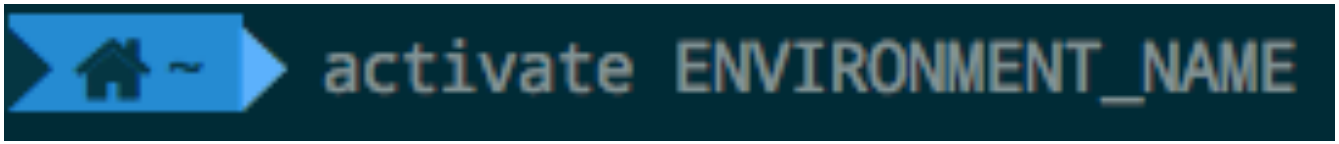


Environment setup

Setting up virtualenv

- Activate the created virtual environment:
  - On Mac/Linux

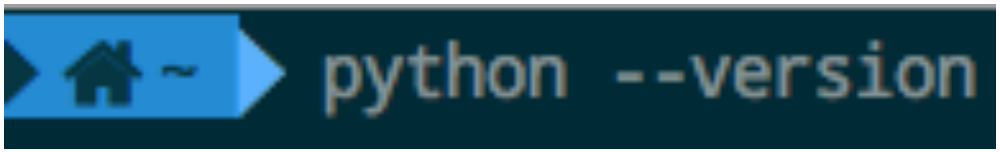
  - On Windows



Environment setup

Setting up virtualenv

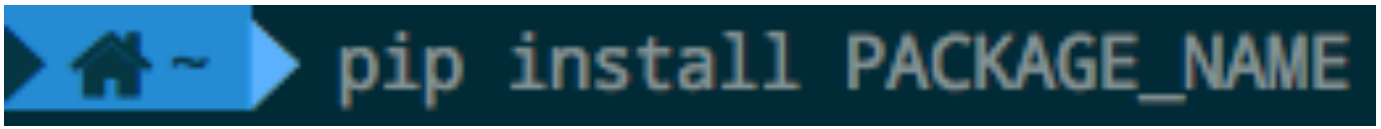
- Check if version is correct (3.6.0):



Environment setup

Setting up virtualenv

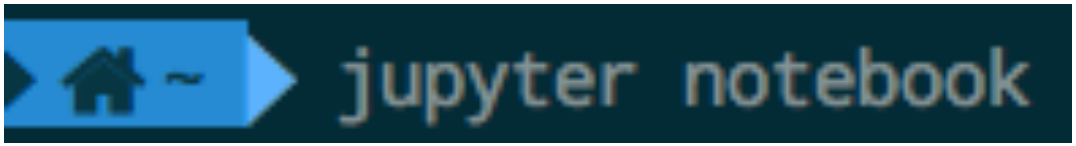
- Install jupyter packages:
  - ipython
  - jupyter



Environment setup

Setting up virtualenv

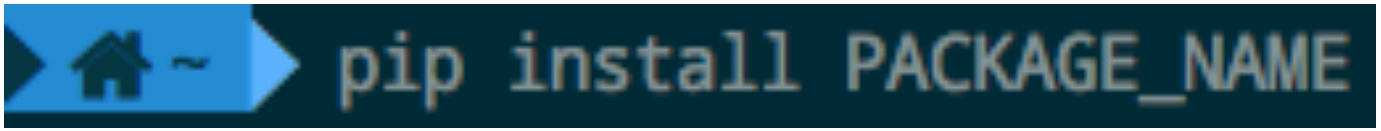
- To launch jupyter notebooks:



Environment setup

Setting up virtualenv

- Install additional packages:
  - numpy
  - scipy
  - Pillow
  - matplotlib



Basic Python

- Basic data types
  - Numbers
    - Types

```
x = 3
print(type(x)) # Prints "<class 'int'>"
```

```
y = 2.5
print(type(y)) # Prints "<class 'float'>"
```

Basic Python

- Basic data types
  - Numbers
    - Operations

```
print(x)        # Prints "3"
print(x + 1)    # Addition; prints "4"
print(x - 1)    # Subtraction; prints "2"
print(x * 2)    # Multiplication; prints "6"
print(x ** 2)   # Exponentiation; prints "9"
```

Basic Python

- Basic data types
  - Numbers
    - Operations

```
x += 1
print(x)   # Prints "4"
x *= 2
print(x)   # Prints "8"
```

Basic Python

- Basic data types
  - Booleans

```
t = True
f = False
print(type(t)) # Prints "<class 'bool'>"
print(t and f) # Logical AND; prints "False"
print(t or f)  # Logical OR; prints "True"
print(not t)   # Logical NOT; prints "False"
print(t != f)  # Logical XOR; prints "True"
```

Basic Python

- Basic data types
  - Strings

```
hello = 'hello'     # String literals can use single quotes
world = "world"     # or double quotes; it does not matter.
print(hello)        # Prints "hello"
print(len(hello))   # String length; prints "5"
hw = hello + ' ' + world  # String concatenation
print(hw)  # prints "hello world"
```

Basic Python

- Basic data types
  - Strings

```
hw12 = '%s %s %d' % (hello, world, 12)  # sprintf style
print(hw12)  # prints "hello world 12"
```

Basic Python

- Containers
  - Lists

```
xs = [3, 1, 2]    # Create a list
print(xs, xs[2])  # Prints "[3, 1, 2] 2"
print(xs[-1])     # Negative indices count from the end
```

Basic Python

- Containers
  - Lists

```
xs[2] = 'foo'     # Lists can contain elements of different types
print(xs)         # Prints "[3, 1, 'foo']"
xs.append('bar')  # Add a new element to the end of the list
print(xs)         # Prints "[3, 1, 'foo', 'bar']"
x = xs.pop()      # Remove and return the last element of the list
print(x, xs)      # Prints "bar [3, 1, 'foo']"
```

Basic Python

- Containers
  - Lists Slicing

```
nums = list(range(5))
print(nums)
print(nums[2:4])
print(nums[2:])
print(nums[:2])
print(nums[:])
print(nums[:-1])
nums[2:4] = [8, 9]
print(nums)
```

```
[0, 1, 2, 3, 4]
[2, 3]
[2, 3, 4]
[0, 1]
[0, 1, 2, 3, 4]
[0, 1, 2, 3]
[0, 1, 8, 9, 4]
```

Basic Python

- Containers
  - Lists Loops

```
animals = ['cat', 'dog', 'monkey']
for animal in animals:
    print(animal)
```

Basic Python

- Containers
  - Dictionaries

```
d = {'cat': 'cute', 'dog': 'furry'}  # Create a new dictionary with some data
print(d['cat'])       # Get an entry from a dictionary; prints "cute"
print('cat' in d)     # Check if a dictionary has a given key; prints "True"
d['fish'] = 'wet'     # Set an entry in a dictionary
print(d['fish'])      # Prints "wet"
```

Basic Python

- Containers
  - Dictionaries Loops

```
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal in d:
    legs = d[animal]
    print('A %s has %d legs' % (animal, legs))
```

```
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal, legs in d.items():
    print('A %s has %d legs' % (animal, legs))
```

Basic Python

- Containers
  - Sets

```
animals = {'cat', 'dog'}
print('cat' in animals)   # Check if an element is in a set,
print('fish' in animals)  # prints "False"
animals.add('fish')       # Add an element to a set
print('fish' in animals)  # Prints "True"
```

Basic Python

- Functions

```
def sign(x):
    if x > 0:
        return 'positive'
    elif x < 0:
        return 'negative'
    else:
        return 'zero'

for x in [-1, 0, 1]:
    print(sign(x))
# Prints "negative", "zero", "positive"
```

Basic Python

- Classes

```
class Greeter(object):

    # Constructor
    def __init__(self, name):
        self.name = name  # Create an instance variable

    # Instance method
    def greet(self, loud=False):
        if loud:
            print('HELLO, %s!' % self.name.upper())
        else:
            print('Hello, %s' % self.name)
```

Basic Python

- Classes

```
g = Greeter('Fred')
g.greet()
g.greet(loud=True)
```

```
Hello, Fred
HELLO, FRED!
```

NumPy

- High-performance multidimensional array object and tools
- More efficient and compact than lists

NumPy

- Arrays

```
import numpy as np

a = np.array([1, 2, 3])   # Create a rank 1 array
print(type(a))            # Prints "<class 'numpy.ndarray'>"
print(a.shape)            # Prints "(3,)"
print(a[0], a[1], a[2])   # Prints "1 2 3"
a[0] = 5                  # Change an element of the array
print(a)                  # Prints "[5, 2, 3]"
```

NumPy

- Arrays creation

```
a = np.zeros((2,2))   # Create an array of all zeros
print(a)              # Prints "[[ 0.  0.]
                      #          [ 0.  0.]]"

b = np.ones((1,2))    # Create an array of all ones
print(b)              # Prints "[[ 1.  1.]]"
```

NumPy

- Arrays creation

```
c = np.full((2,2), 7)  # Create a constant array
print(c)               # Prints "[[ 7.  7.]
                       #          [ 7.  7.]]"

d = np.eye(2)          # Create a 2x2 identity matrix
print(d)               # Prints "[[ 1.  0.]
                       #          [ 0.  1.]]"
```

NumPy

- Array indexing

```
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# Use slicing to pull out the subarray consisting of the first 2 rows
# and columns 1 and 2; b is the following array of shape (2, 2):
# [[2 3]
#  [6 7]]
b = a[:2, 1:3]
```

NumPy

- Array indexing - mutating

```
# Create a new array from which we will select elements
a = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
```

```
# Create an array of indices
b = np.array([0, 2, 0, 1])
```

```
# Mutate one element from each row of a
a[np.arange(4), b] += 10
```

NumPy

- Boolean array indexing

```
a = np.array([[1,2], [3, 4], [5, 6]])
```

```
bool_idx = (a > 2)
```

NumPy

- Datatypes

```
x = np.array([1, 2])   # Let numpy choose the datatype
print(x.dtype)         # Prints "int64"

x = np.array([1.0, 2.0])   # Let numpy choose the datatype
print(x.dtype)             # Prints "float64"

x = np.array([1, 2], dtype=np.int64)   # Force a particular datatype
print(x.dtype)                         # Prints "int64"
```

NumPy

- Array math

```
x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

# Elementwise sum; both produce the array
# [[ 6.0  8.0]
#  [10.0 12.0]]
print(x + y)
print(np.add(x, y))
```

NumPy

- Array math

```
x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])
```

```
# Matrix / vector product; both produce the rank 1 array [29 67]
print(x.dot(v))
print(np.dot(x, v))
```

SciPy

- Linear algebra and numerical algorithms
- Scientific and engineering applications

SciPy

- Image operations

```
from scipy.misc import imread, imsave, imresize

# Read an JPEG image into a numpy array
img = imread('assets/cat.jpg')
print(img.dtype, img.shape)  # Prints "uint8 (400, 248, 3)"
```

```
img_tinted = img * [1, 0.95, 0.9]

# Resize the tinted image to be 300 by 300 pixels.
img_tinted = imresize(img_tinted, (300, 300))

# Write the tinted image back to disk
imsave('assets/cat_tinted.jpg', img_tinted)
```

SciPy

- Image operations

SciPy

- Distance between points

```
import numpy as np
from scipy.spatial.distance import pdist, squareform

# Create the following array where each row is a point in 2D space:
# [[0 1]
#  [1 0]
#  [2 0]]
x = np.array([[0, 1], [1, 0], [2, 0]])
print(x)
```

```
d = squareform(pdist(x, 'euclidean'))
print(d)
```

Matplotlib

- Plotting library
- High quality, several kinds of plots

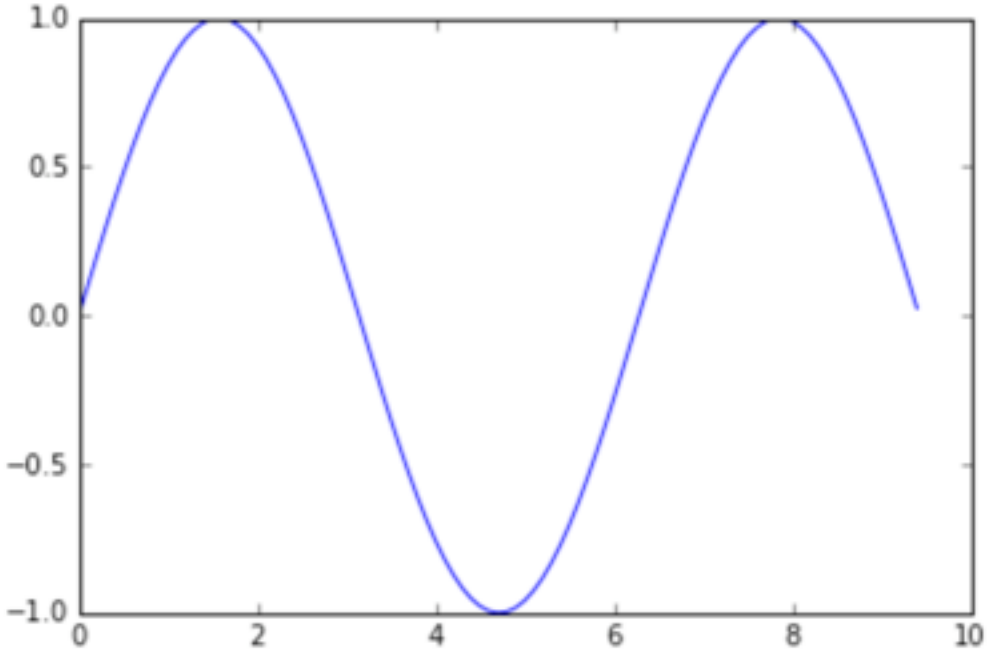
Matplotlib

- Plotting

```
import numpy as np
import matplotlib.pyplot as plt

# Compute the x and y coordinates for points on a sine curve
x = np.arange(0, 3 * np.pi, 0.1)
y = np.sin(x)

# Plot the points using matplotlib
plt.plot(x, y)
plt.show()  # You must call plt.show() to make graphics appear.
```

Matplotlib

- Plotting



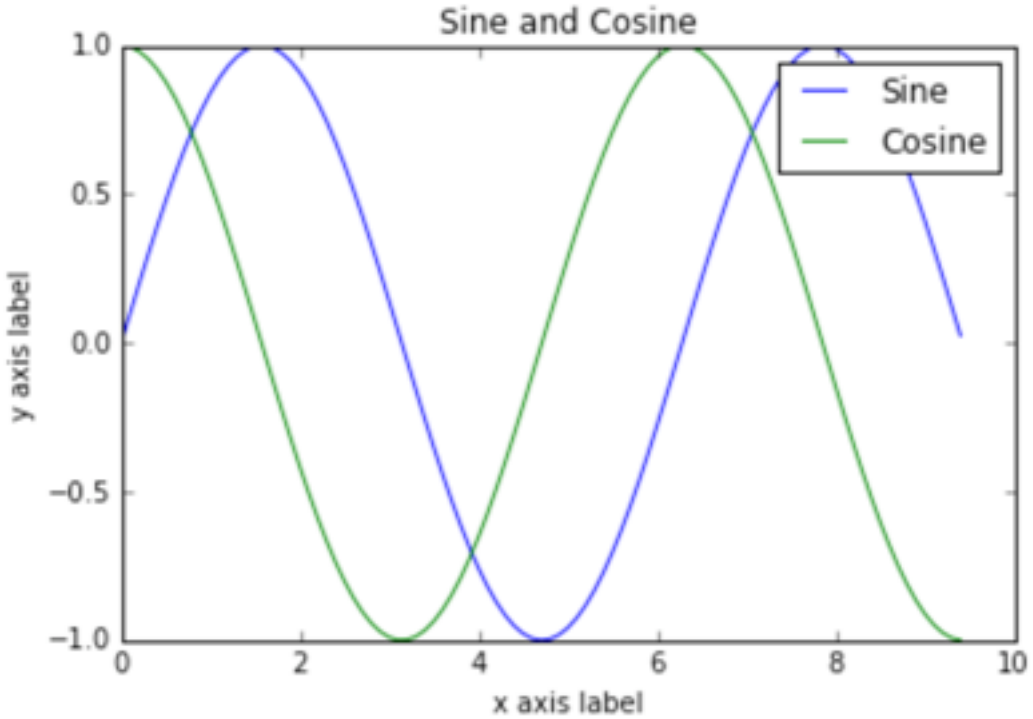
Matplotlib

- Plotting

```
import numpy as np
import matplotlib.pyplot as plt

# Compute the x and y coordinates for points on sine and cosine curves
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

# Plot the points using matplotlib
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend(['Sine', 'Cosine'])
plt.show()
```

Matplotlib

- Plotting



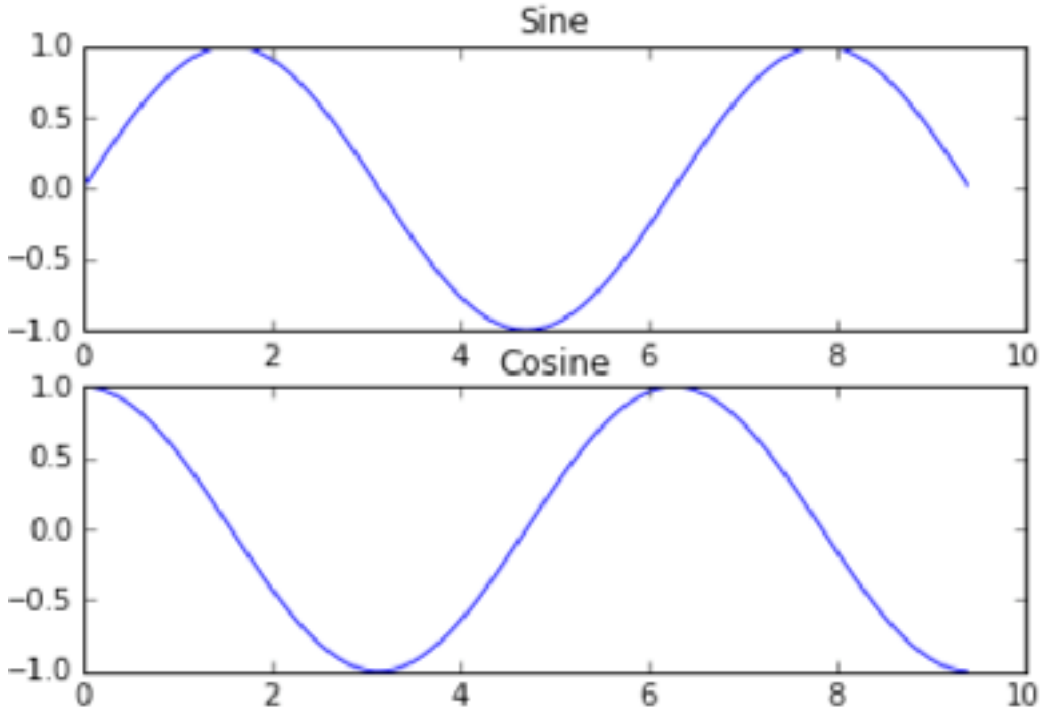
Matplotlib

- Subplots

```
# Set up a subplot grid that has height 2 and width 1,
# and set the first such subplot as active.
plt.subplot(2, 1, 1)

# Make the first plot
plt.plot(x, y_sin)
plt.title('Sine')

# Set the second subplot as active, and make the second plot.
plt.subplot(2, 1, 2)
plt.plot(x, y_cos)
plt.title('Cosine')

# Show the figure.
plt.show()
```

Matplotlib

- Subplots



Assignment

- Optional, but good for getting hands on
- Solution will be handed online

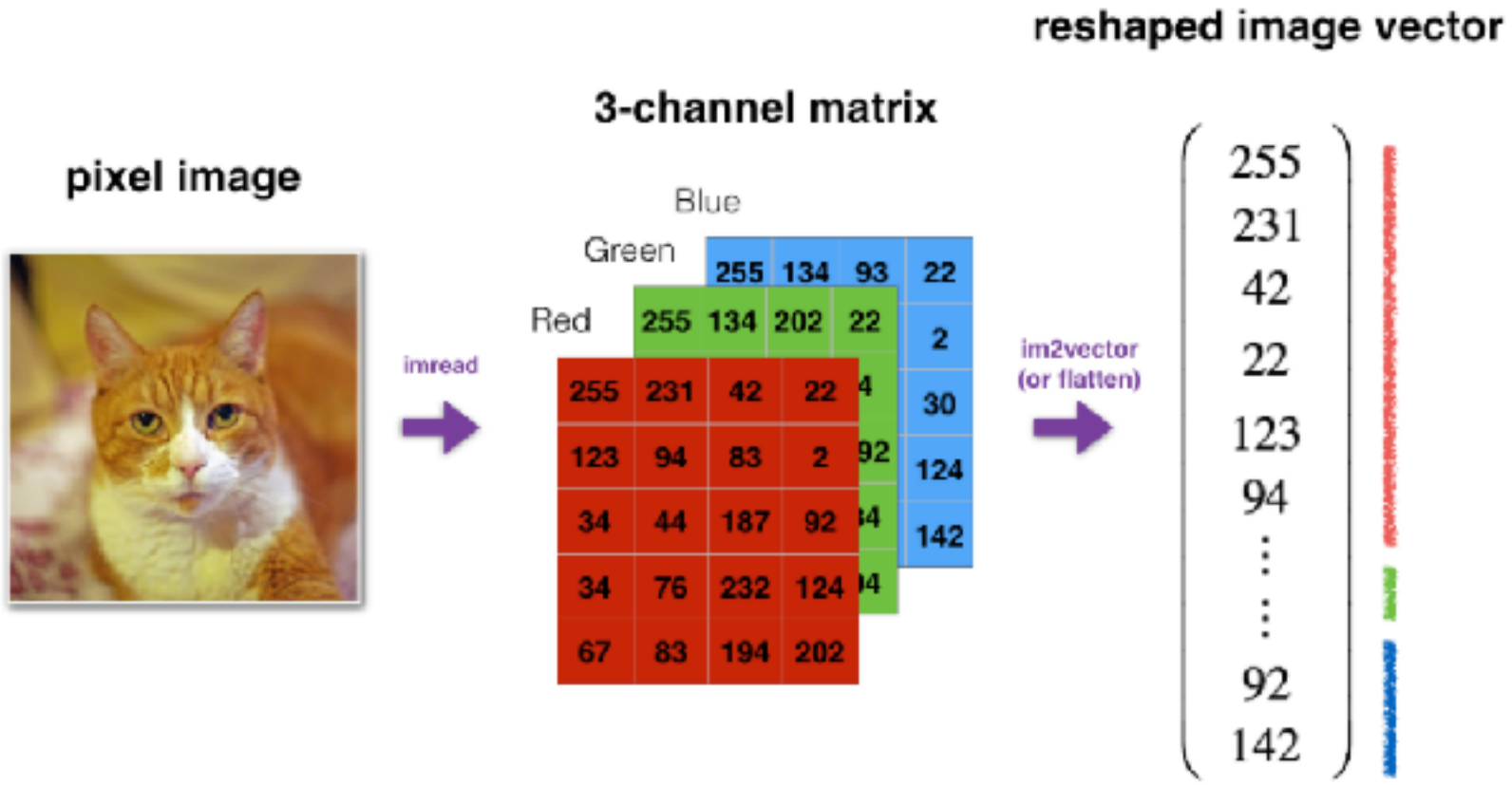
Assignment

- Common numpy functions in deep learning are np.shape and np.reshape()
  - X.shape is used to get the shape (dimension) of a matrix/vector X
  - X.reshape(...) is used to reshape X into some other dimension

Assignment

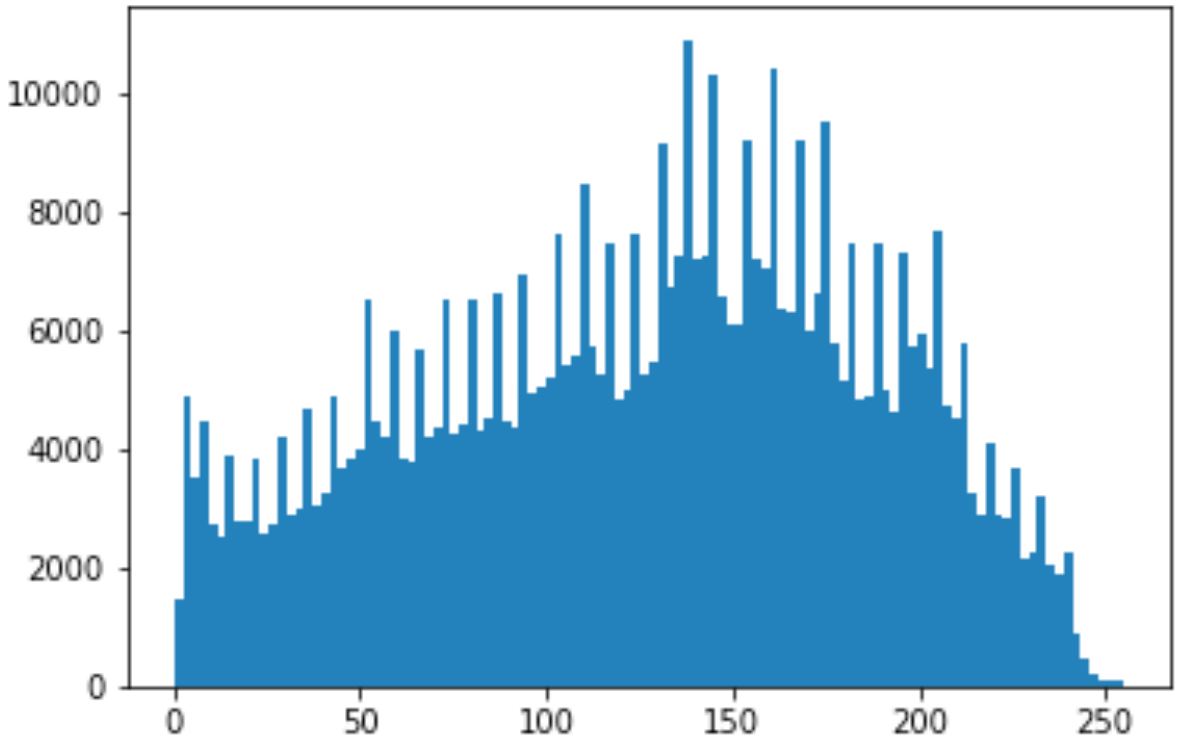
- Image are represented by a 3D array of shape ***(length,height,depth=3)***
- Reading image as input of algorithm you convert it to a vector of shape ***(length∗height∗3,1)***
- Reshape 3D array into 1D vector.

Assignment


pixel image
imread
3-channel matrix
Blue
Green
Red
255 134 93 22
255 134 202 22
255 231 42 22
123 94 83 2
34 44 187 92
34 76 232 124
67 83 194 202
2
30
124
142
im2vector
(or flatten)
reshaped image vector
255
231
42
22
123
94
92
142

Assignment

- Histogram: Tonal distribution in image



Assignment

- **Exercise:**
  - Reshape image array using NumPy
    - Input of shape ***(length, height, 3)***
    - Reshape into vector ***(length*height*3, 1)***
  - Use matplotlib to plot the vector as a histogram

References

- Stanford's cs231n course
- NumPy Cheatsheet
- SciPy Cheatsheet
- Matplotlib Cheatsheet